ISEF Project Roadmap: Physics-Informed Neural Networks for Orbital Propagation

Project Goal: *Develop a High-Fidelity Orbital Propagator using Physics-Informed Neural Networks (PINNs) to outperform standard AI models in predicting satellite trajectories.*

Phase 1: The Foundation (Data Generation)

Goal: *Create the "Ground Truth" dataset. We cannot train an AI without knowing the correct answers first.*

👤 *Role: The Mathematician*

- ***Verify the Constants:*** *Confirm we are using the correct gravitational parameter for Earth ($\mu = 3.986 \times 10^5 \text{ km}^3/\text{s}^2$) and Earth's radius ($R_E = 6378 \text{ km}$).*
- ***The Equation Check:*** *Verify that the Two-Body Equation of Motion is correctly formulated for the code:*
  $$\ddot{\mathbf{r}} = -\frac{\mu}{||\mathbf{r}||^3}\mathbf{r}$$
- ***Units:*** *Ensure all data is consistent (km and seconds). Neural networks struggle with huge numbers, so we may need to normalize this later (e.g., divide distances by Earth's radius).*

💻 *Role: The Coder*

- ***Environment Setup:*** *Install `numpy`, `scipy`, `torch`, `matplotlib`.*
- ***Script Implementation:*** *Write and run the `generate_data.py` script (provided in chat).*
- ***Data Generation:***
  - *Simulate a Low Earth Orbit (LEO) for 3 orbital periods.*
  - *Save the output `(t, x, y, z, vx, vy, vz)` to `orbital_data.npy`.*
- ***Sanity Check:*** *Plot the generated data in 3D. If the orbit isn't a closed ellipse, do not proceed.*

Phase 2: The "Straw Man" (Baseline Model)

Goal: *Build a "dumb" standard neural network that fails. This establishes a baseline to prove why the PINN is necessary.*

👤 *Role: The Mathematician*

- ***Define the Metric:*** *specific strictly how we measure "failure."*
    - *Metric: Root Mean Square Error (RMSE) of position over time.*
- ***Analyze the Drift:*** *When the standard model fails, does it break Conservation of Energy? Calculate the specific energy at the point of failure to prove it violates physics.*

💻 *Role: The Coder*

- ***Build the MLP:*** *Create a standard Multi-Layer Perceptron in PyTorch.*
    - *Input: Time (t).*
    - *Hidden: 3 layers $\times$ 64 neurons (Tanh activation).*
    - *Output: Position (x, y, z).*
- ***The "Black Box" Training:*** *Train this model on the first 80% of the orbit data using only* ***MSE Loss*** *(Mean Squared Error).*
- ***The Failure Plot:*** *Ask the model to predict the final 20% of the orbit. Plot the prediction vs. the ground truth. It should diverge significantly.* ***Save this plot for the science fair board.***

Phase 3: The PINN (Physics-Informed Core)

Goal: *The core innovation. Teaching the AI the laws of physics using a custom loss function.*

👤 *Role: The Mathematician*

- ***Derive the Residual:*** *Write the exact loss function equation for the code.*
    - *Residual (f): The difference between the Neural Network's second derivative and Newton's law.*
    - $$f = \frac{d^2\hat{y}}{dt^2} + \frac{\mu}{||\hat{y}||^3}\hat{y}$$
    - *(Where $\hat{y}$ is the network's predicted position).*
- ***Hyperparameter Tuning:*** *Determine the weight (λ) for the physics loss. Start with $\lambda = 1.0$. If the model ignores data, lower it. If it ignores physics, raise it.*

💻 *Role: The Coder*

- ***Implement Autograd:*** *Use `torch.autograd.grad` to calculate the first derivative (velocity) and second derivative (acceleration) of the network output with respect to the input time (t).*
- ***Custom Loss Function:***
    - `Loss = MSE_Data + (lambda * MSE_Physics_Residual)`
- ***Training:*** *Retrain the model.*
- ***Validation:*** *Plot the new prediction for the final 20% of the orbit. It should now hug the true line much closer than the Phase 2 model.*

Phase 4: High-Fidelity Upgrade (J2 Perturbation)

Goal: *Increasing complexity to "Research Level." accounting for Earth's non-spherical shape.*

👤 *Role: The Mathematician*

- ***The J2 Formula:*** *Provide the Cartesian acceleration equations for the J2 perturbation (Earth's oblateness).*
  - *Note: This adds terms involving z^2 and r^2 to the acceleration equation.*
- ***Precession Analysis:*** *The orbit should now rotate (precess) slowly over time. Verify this behavior is mathematically expected.*

💻 *Role: The Coder*

- ***Update Physics Engine:*** *Update the Phase 1 generation script to include J2 acceleration so we have new "Ground Truth" data.*
- ***Update Loss Function:*** *Add the J2 terms to the PINN's physics loss function.*
- ***Long-Term Test:*** *Train on 1 orbit, predict 5 orbits. Show that the PINN captures the "wobble" (precession) that a standard neural network misses completely.*

Phase 5: Metrics & Analysis

Goal: *Generating the numbers and graphs for the presentation.*

👤 *Role: The Mathematician*

- ***Hamiltonian Check:*** *Calculate the Total Energy ($H = \text{Kinetic} + \text{Potential}$) for every predicted point.*
  - *Plot H over time. The PINN should be nearly flat (conserved), while the standard NN fluctuates wildly.*
- ***Error Tables:*** *Create a table comparing RMSE for "Standard NN" vs "PINN" at $t=100s$, $t=1000s$, and $t=5000s$.*

💻 *Role: The Coder*

- ***The "Money Shot" Plot:*** *Create a high-resolution 3D plot showing:*
  1. *Ground Truth (Solid Blue Line)*
  2. *Standard NN Prediction (Dotted Red Line - Diverging)*
  3. *PINN Prediction (Dashed Green Line - Accurate)*
- ***Code Cleanup:*** *Organize the code into a clean GitHub repository with a `README.md` explaining how to run it.*

Phase 6: Deliverables

1. ***Codebase:*** *GitHub Repo with `generate_data.py`, `train_pinn.py`, and `plotting.py`.*
2. ***Visuals:***
    - *Orbit Failure Plot (Phase 2).*
    - *Physics Loss Convergence Graph (Phase 3).*
    - *Energy Conservation Graph (Phase 5).*
3. ***Abstract:*** *A 250-word summary highlighting that PINNs reduce orbital propagation error by $X\%$ compared to standard deep learning methods while enforcing physical laws.*